# Product Derivation Process and Agile Approaches: Exploring the Integration Potential

Padraig O'Leary, Muhammad Ali Babar, Steffen Thiel, Ita Richardson

Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland.
{padraig.oleary, malibaba, steffen.thiel, ita.richardson}@lero.ie

**Abstract.** Software Product Lines (SPL) and Agile practices have emerged as new paradigms for developing software. Both approaches share common goals; this provides the motivation for exploring the possibilities of integrating these two approaches. However, there has been little research on identifying the opportunities and challenges of such integration. We have been researching the potential of integrating Agile approaches in one of the key SPL process areas, product derivation. In this paper, we identify the similarities and differences between Agile and SPL and propose the idea of introducing Agile practices to improve the product derivation process.

## 1. Introduction

Both Agile and Software Product Lines (SPL) software development paradigms are being promoted as means of reducing time to market, increasing productivity, improving quality and gaining cost effectiveness and efficiency [1] of software development efforts. Furthermore, both approaches assume that requirement changes will occur and can be managed[1]. These shared common goals by Agile and SPL open the possibilities of introducing Agile practices into SPL activities. There are, however, several challenges to integrating Agile approaches in SPL due to the differences that exist in the philosophies of both approaches such as design and change management strategies [1, 2]. Moreover, Agile approaches do not purpose to develop flexible artefacts for reuse [2, 3] or develop the documentation for maintenance and evolution required by SPL [3]. Our research in SPL is aimed at improving the Product Derivation (PD) process, which purports to develop new products by utilizing core assets of a SPL such as feature models and architecture models [4]. We decided to concentrate on PD as it is considered the most important and challenging SPL activity. We believe that any successful effort to improve PD can make SPL substantially more effective and efficient. However, despite the increasing trend of using Agile approaches for developing large scale systems, there has been little research conducted on the use of Agile approaches in the context of SPL or more specifically PD. This paper presents our initial effort to study the opportunities and challenges of using Agile in SPL.

The next section describes the key concepts of SPL and Agile practices. Section three compares SPL and Agile. Section four presents a generic product derivation framework. Section five discusses the potential of using Agile in PD. This paper finishes in section 6 with a summary and a description of our future work.

## 2. Background and Motivation

In the following sections, we discuss the main concepts of Agile practices and SPL that underpin our proposal for integrating the two paradigms.

### 2.1 Software Product Lines

A SPL is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [5]. The SPL approach makes a distinction between domain engineering, where a common platform for an arbitrary number of products is designed and realised, and application engineering, where a product is derived [6]. The separation into domain engineering and application engineering allows the development of software artefacts which are shared among the products within that domain. These shared artefacts become separate entities in their own right, subscribing to providing shared functionality across multiple products. It is during application engineering that the individual products within a product line are constructed. The products are constructed using a number of shared software artefacts created during domain engineering. The process of creating these individual products using the shared artefacts is known as the product derivation process.

### 2.2 Agile Practices

Agile Practices have recently gained popularity among large numbers of companies as a mechanism for reducing costs and increasing ability to handle change in dynamic market conditions. Researchers and practitioners have proposed several software development approaches based on the principles of the Agile manifesto [7] [8]. The Agile manifesto allows for changing requirements throughout the development cycle and stresses collaboration between software developers and customers, and early product delivery. One of the central principles of the Agile approach is the focus on people. People coupled with effectiveness and manoeuvrability are considered the primary drivers of project success [9].

The Agile manifesto provides the framework for two of the best known Agile practices, eXtreme Programming (XP) and Scrum. Although they are based on a common guideline defined by the Agile manifesto, they vary in focus and presentation, XP emphasises technical elements of the development lifecycle; while Scrum concentrates on the project management elements.

## 3. Comparing Agile and SPL Paradigms

Agile Practices and SPL share common goals such as improving productivity, reducing time to market, decreasing development costs and increasing customer satisfaction [1, 10, 11]. Furthermore, both Agile and SPL can accommodate requirements change during the development process [1, 10, 11]. Additionally, adaptation of Agile or SPL requires change in the development process and the artefacts and roles associated with this process. Despite having similar goals, both Agile and SPL have different mechanics and logistics for achieving their respective

goals. For example, Agile practices and SPL have different strategies for change management - Agile practices handle change through incremental development while SPL uses its core assets[1, 2]. Agile practices do not advocate the development of flexible artefacts for reuse such as core assets in SPL [2, 3].

SPL aims to fulfil the requirements of several customers rather than concentrating on meeting the individual needs of a particular customer [3]. Customer involvement is much more intensive in Agile practices where developers work closely with customers on daily basis. Agile and SPL paradigms also differ in terms of role and philosophy of software design. Agile approaches do not emphasis the importance of design and documentation; rather Agile approaches advocate implementing required functionality and then documenting the prepared code by reconstruction and refactoring. On the other hand, design and documentation are very vital activities in SPL as core assets need to be appropriately documented to support their reusability [2].

These differences in the fundamental philosophies of the two development paradigms are likely to make any effort to integrate Agile and SPL quite challenging. However, there appears to be many potential benefits of successfully integrating Agile and SPL. An Agile SPL approach may have the potential of supporting the effective and efficient development of much larger families of products than a pure SPL approach [1]. For example, Agile testing techniques such as frequent, early testing and with a high emphasis on keeping code error free would benefit SPL companies significantly [1]. Moreover, the Agile project management approach Scrum can be used to manage the product derivation process and development practices like XP or test-driven development can be used for developing core and/or product components.

## 4. A Product Derivation Framework

In this section, we briefly describe a framework that we have developed to support the product derivation process of SPL. We have developed this framework based on the results of an extensive literature review, lengthy discussions with SPL practitioners and researchers, and reviewing the documentation on product derivation activities by our industrial collaborators. This framework consists of four main steps, which are:

1. Impact Analysis;
2. Reusability Analysis;
3. Component Development; and
4. Product Integration and Validation.

Figure 1 provides a diagrammatic representation of the product derivation framework and the following paragraphs provide a brief description of each of the four steps. Impact analysis (step 1) is aimed at gathering product specific requirements based on customer requirements and negotiation with the Platform Team. Reusability Analysis (step 2) purports to create a partial product configuration based on the product specific requirements and by using the available core assets. During Component Development (step 3), new components are developed (if required) and existing components are adapted to satisfy requirements which could not be satisfied by configuring existing core assets. Finally, Product Integration and

Validation (step 4) aims to integrate the core asset configuration and newly developed components and to validate the integration by performing appropriate testing procedures.

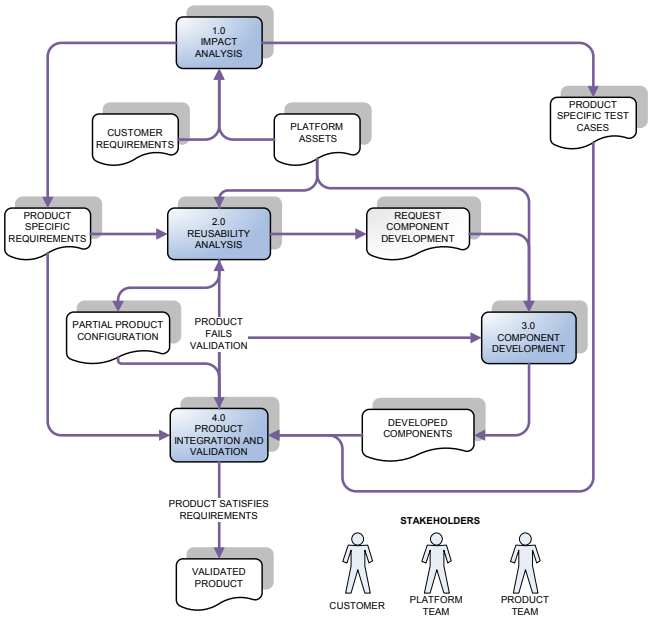


**Figure 1: An Overview of the Product Derivation Framework**

## 5. Integrating Agile Practices in Product Derivation Framework

In this section we discuss the potential for integrating Agile practices in our Product Derivation framework (presented in Section 4). Having analyzed the SPL's two major phases, domain engineering and application engineering, it is the application engineering phase that is expected to benefit the most from incorporating Agile practices [2, 3].

For example, during Impact Analysis, the product team can benefit from applying the planning game practice from the XP methodology for gathering and negotiating product specific requirements. In the planning game, a customer priorities the requirements and the developers estimate the effort required to satisfy those requirements. The end dates of iterations are specified and requirements are allocated to specific iterations based on their priority [2]. Noor et al. also identifies the use of Agile practices to support collaborative work with stakeholders [12], assisting the product team in Impact Analysis.

The identification of Product Derivation iterations is a key aspect of deriving high quality, customer satisfying products, according to Carbon et al.[2] with a SPL, an organisation is capable of producing a first version of a product for a specific customer, including the core functionality, quicker than other software development methods. Because of the approved quality of the reusable assets, the customer directly gets a high quality product that can be used and evaluated to give feedback. In further

iterations, new functionality can be added to the scope of the product line or product specific features can be implemented [2].

According to Kurmann, automation is the key aspect for an Agile software product line development. Most important, the SPL product process has to be fully automated. Automation and iterative development would also facilitate another Agile Principle "*Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*" [8], as iterative development would allow product teams implement changing customer requirements quite late in the development lifecycle.

Unit and component tests as well as smoke tests for each software product should be automated [10]. In step three, Component Development, the Unit and component tests would become automated, while in step four, Product Integration and Validation, integration and System testing could be partially automated.

For step three, Component Development, the Agile principle of "early and continuous delivery of valuable software" identified in the Agile manifesto [8] ties in well for two reasons. Firstly, instead of delaying development while the CCB scope required changes and the platform team implement all requested platform changes, the product team simply implement changes at the product level. The platform team can subsequently mine any changes from the product if there is reuse potential. Secondly, pair programming can be used to implement and review any changes at the product level [10]. This helps to produce better quality product code and as a result better quality for mined platform code.

## 6. Summary and Future Work

Our research is aimed at making SPL more effective and efficient. To achieve this objective, we have been concentrating on improving the product derivation process by developing and empirically assessing different approaches and tools. In this paper, we have explored the potential for integrating Agile practices in the PD process. We have discussed the similarities and differences between the two approaches. We have identified a set of Agile practices that have potential for integration into the PD process. We have also identified activities within our product derivation framework where Agile practices could be introduced.

We believe that incorporating Agile practices into the generic framework can bring a balance between agility and formalism during the product derivation process. A combination of Agile and SPL is expected to create a leaner but more disciplined process [10]. We are rigorously assessing the effectiveness and efficiency of the framework with several of our industrial collaborators, which have adopted the SPL approach for developing software intensive systems. These industry based assessments will help us to identify not only areas of improvement in our framework but also investigate the logistics and effects of introducing Agile practices in SPL.

Our future work includes an ongoing investigation into the strengths and weaknesses of Agile and SPL approaches in combination [2], validation of an Agile SPL approach [1] and the expected return on investment from a combined approach [2].

## 7. Acknowledgements

This work is partially supported by IRCSET under grant no. RS/06/167 and Science Foundation Ireland under grant number 03/CE2/I303-1.

## 8. References

1. Tian, K. and K. Cooper, *Agile and Software Product Line Methods: Are They So Different?*, in *1st International Workshop on Agile Product Line Engineering*. 2006: Baltimore, Maryland, USA.
2. Carbon, R., et al. *Integrating Product Line Engineering and Agile Methods: Flexible Design Up-front vs. Incremental Design.* in *In Proceedings of the 1st International Workshop on Agile Product Line Engineering (APLE'06)* 2006. Kyoto, Japan.
3. Navarrete, F., P. Botella, and F. Xavier. *An Approach to Reconcile the Agile and CMMI Context in Product Line Development*. in *In Proceedings of the 1st International Workshop on Agile Product Line Engineering (APLE'06)* 2006. Kioto, Japan.
4. Deelstra, S., M. Sinnema, and J. Bosch, *Product Derivation in Software Product Families, A Case Study.* Journal of Systems and Software, 2005. 74: p. 173-194.
5. Clements, P. and L. Northrop, *Software Product Lines: Practices and Patterns*. 2002, Reading, MA: Addison Wesley.
6. Hotz, L., A. Gunter, and T. Krebs, *A Knowledge-based Product Derivation Process and some Ideas how to Integrate Product Development*, in *Proc. of Software Variability Management Workshop*. 2003: Groningen, The Netherlands.
7. Abrahamsson, P., et al. *New Directions on Agile Methods: A Comparative Analysis*. in *ICSE 2003*. 2003. Portland, USA.
8. Beck, K., et al. *Manifesto for Agile Software Development*. 2001 [cited 2002 22.3.2002]; Available from: http://AgileManifesto.org.
9. Highsmith, J. and A. Cockburn, *Agile Software Development: The Business of Innovation.* IEEE Computer, 2001. 34(9): p. 120-122.
10. Kurmann, R. *Agile SPL-SCM Agile Software Product Line Configuration and Release Management*. in *In Proceedings of the 1st International Workshop on Agile Product Line Engineering (APLE'06)*. 2006. Kioto, Japan.
11. Trinidad, P., et al. *Explanations for Agile Feature Models*. in *In Proceedings of the 1st International Workshop on Agile Product Line Engineering (APLE'06)*. 2006. Kyoto, Japan.
12. Noor, M., R. Rabiser, and P. Grunbacher, *A Collaborative Approach for Reengineering-based Product Line Scoping*, in *In Proceedings of the 1st International Workshop on Agile Product Line Engineering (APLE'06)*. 2006: Kyoto, Japan.